数论

容斥原理

例题

CF1900D. Small GCD

首先排序，然后枚举 a_i 作为三元组中间元素，方案数乘上 $n-i$ （a_k 的个数）。

c_x 表示 i 前面因数为 x 的 j 的个数。
g_x 表示 $gcd(i,j)=(x$的倍数$)$ 的 (i,j) 对数
f_x 表示 $gcd(i,j)=x$ 的 (i,j) 对数
$f_x=g_x-f_{2x}-f3x-\dots$
x 的贡献：$f_x*x*(n-i)$

```cpp
#include <bits/stdc++.h>
#define endl "\n"
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
typedef tuple<char,char,char,char> node;
const int INF = 0x3f3f3f3f;
const int N = 100000;

vector< vector<int> > fac(N + 1);

void init(int n) {
	for(int i = 1; i <= n; i++) {
		for(int j = i; j <= n; j += i) {
			fac[j].push_back(i);
		}
	}
}

void solve() {
	int n;
	cin >> n;
	vector<int> a(n + 1);
	vector<LL> c(N + 1, 0), f(N + 1, 0);
	for(int i = 1; i <= n; i++) {
		cin >> a[i];
	}
	sort(a.begin() + 1, a.end());
	for(int i = 1; i <= n; i++) {
		for(int x : fac[a[i]]) {
			f[x] += c[x] * (n - i);
			c[x]++;
		}
	}
	LL ans = 0;
	for(int i = N; i; i--) {
		for(int j = i + i; j <= N; j += i) {
			f[i] -= f[j];
		}
		ans += f[i] * i;
```

```
        }
        cout << ans << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        init(N);
        int t = 1;
        cin >> t;
        while(t--) solve();
        return 0;
}
```

FFT

例题

Acwing3122. 多项式乘法——模板

```
#include <bits/stdc++.h>
#define endl "\n"
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 300010;
const double PI = acos(-1);

int bit, tot;
int rev[N];
struct Complex {
        double x, y;
        Complex operator + (const Complex &t) const {
                return {x + t.x, y + t.y};
        }
        Complex operator - (const Complex &t) const {
                return {x - t.x, y - t.y};
        }
        Complex operator * (const Complex &t) const {
                return {x * t.x - y * t.y, x * t.y + y * t.x};
        }
}a[N], b[N];

void fft(Complex a[], int inv) {
        for(int i = 0; i < tot; i++) {
                if(i < rev[i]) swap(a[i], a[rev[i]]);
        }
        for(int mid = 1; mid < tot; mid <<= 1) {
                auto w1 = Complex{cos(PI / mid), inv * sin(PI / mid)};
                for(int i = 0; i < tot; i += mid * 2) {
                        auto wk = Complex{1, 0};
                        for(int j = 0; j < mid; j++, wk = wk * w1) {
```

```
                                auto x = a[i + j], y = wk * a[i + j + mid];
                                a[i + j] = x + y, a[i + j + mid] = x - y;
                        }
                }
        }
}

void solve() {
        int n, m;
        cin >> n >> m;
        for(int i = 0; i <= n; i++) cin >> a[i].x;
        for(int i = 0; i <= m; i++) cin >> b[i].x;
        while((1 << bit) < n + m + 1) bit++;
        tot = 1 << bit;
        for(int i = 0; i < tot; i++) {
                rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
        }
        fft(a, 1), fft(b, 1);
        for(int i = 0; i < tot; i++) a[i] = a[i] * b[i];
        fft(a, - 1);
        for(int i = 0; i <= n + m; i++) {
                cout << (int)(a[i].x / tot + 0.5) << ' ';
        }
        cout << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) solve();
        return 0;
}
```

图论

圆方树模板

```
void tarjan(int u) { //tarjan 求出点的双连通分量，G 中存储的即为圆方树中的边
        dfn[u] = low[u] = ++cnt;
        stk.push_back(u);
        for(int v : g[u]) {
                if(!dfn[v]) {
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                        if(low[v] == dfn[u]) {
                                sz++;
                                G[n + sz].push_back(u);
                                G[u].push_back(n + sz);
                                int x;
                                do {
                                        x = stk.back(); stk.pop_back();
                                        G[x].push_back(n + sz);
                                        G[n + sz].push_back(x);
                                } while(x != v);
                        }
                } else {
                        low[u] = min(low[u], dfn[v]);
                }
        }
}
```

例题

Acwing2863. 最短路

```
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 12010, M = N * 3;

int n, m, q, cnt, nn;
int dis[N], dep[N], pre[N], fw[N], fe[N];
int fa[N][14], s[N], stot[N];
int dfn[N], low[N];
int A, B;
int h1[N], h2[N], w[M], e[M], ne[M], idx;

void add(int h[], int a, int b, int c) {
        e[idx] = b, ne[idx] = h[a], w[idx] = c, h[a] = idx++;
}

void build_circle(int x, int y, int w) {
        int sum = w;
        nn++;
        for(int k = y; k != x; k = pre[k]) {
                s[k] = sum;
                sum += fw[k];
        }
        s[x] = stot[x] = sum;
        add(h2, x, nn, 0);
        for(int k = y; k != x; k = pre[k]) {
                stot[k] = sum;
                add(h2, nn, k, min(s[k], sum - s[k]));
        }
}

void tarjan(int u, int from) {
        dfn[u] = low[u] = ++cnt;
        for(int i = h1[u]; ~i; i = ne[i]) {
```

```
                int v = e[i];
                if(!dfn[v]) {
                        pre[v] = u, fw[v] = w[i], fe[v] = i;
                        tarjan(v, i);
                        low[u] = min(low[u], low[v]);
                        if(dfn[u] < low[v]) { //u - v 这条边是桥，直接加到圆方树中
                                add(h2, u, v, w[i]);
                        }
                } else if(i != (from ^ 1)) {
                        low[u] = min(low[u], dfn[v]);
                }
        }
        for(int i = h1[u]; ~i; i = ne[i]) {
                int v = e[i];
                if(dfn[u] < dfn[v] && fe[v] != i) { // u -> u 是一个环，v 在环上
                        build_circle(u, v, w[i]);
                }
        }
}

void dfs_lca(int u, int father) {
        dep[u] = dep[father] + 1;
        fa[u][0] = father;
        for(int k = 1; k < 14; k++) {
                fa[u][k] = fa[fa[u][k - 1]][k - 1];
        }
        for(int i = h2[u]; ~i; i = ne[i]) {
                int v = e[i];
                dis[v] = dis[u] + w[i];
                dfs_lca(v, u);
        }
}

int get_lca(int a, int b) {
        if(dep[a] < dep[b]) swap(a, b);
        for(int k = 13; k >= 0; k--) {
                if(dep[fa[a][k]] >= dep[b]) {
                        a = fa[a][k];
                }
        }
        if(a == b) return a;
        for(int k = 13; k >= 0; k--) {
                if(fa[a][k] != fa[b][k]) {
```

```
                        a = fa[a][k];
                        b = fa[b][k];
                }
        }
        A = a, B = b;
        return fa[a][0];
}

void solve() {
        cin >> n >> m >> q;
        memset(h1, -1, sizeof h1);
        memset(h2, -1, sizeof h2);
        nn = n;
        while(m--) {
                int u, v, w;
                cin >> u >> v >> w;
                add(h1, u, v, w);
                add(h1, v, u, w);
        }
        tarjan(1, -1);
        dfs_lca(1, 0);
        while(q--) {
                int u, v;
                cin >> u >> v;
                int lca = get_lca(u, v);
                if(lca <= n) {
                        cout << dis[u] + dis[v] - dis[lca] * 2 << '\n';
                } else {
                        int ans = dis[u] - dis[A] + dis[v] - dis[B];
                        int t = abs(s[A] - s[B]);
                        ans += min(t, stot[A] - t);
                        cout << ans << '\n';
                }
        }
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) {
                solve();
```

```
        }
        return 0;
}
```

[ABC318G] Typical Path Problem

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 200010, M = N * 2;

int n, m, a, b, c, sz;
int dfn[N], low[N], cnt;
int fa[N];
vector<int> g[N], G[M], stk;

void tarjan(int u) {
        dfn[u] = low[u] = ++cnt;
        stk.push_back(u);
        for(int v : g[u]) {
                if(!dfn[v]) {
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                        if(low[v] == dfn[u]) {
                                sz++;
                                G[n + sz].push_back(u);
                                G[u].push_back(n + sz);
                                int x;
                                do {
                                        x = stk.back(); stk.pop_back();
                                        G[x].push_back(n + sz);
                                        G[n + sz].push_back(x);
                                } while(x != v);
                        }
                } else {
                        low[u] = min(low[u], dfn[v]);
                }
        }
}

void dfs(int u, int father) {
        fa[u] = father;
```

```
        for(int v : G[u]) {
                if(v != father) {
                        dfs(v, u);
                }
        }
}

bool check(int u) {
        while(u != c) {
                if(u > n) {
                        for(int v : G[u]) {
                                if(v == b) return true;
                        }
                }
                u = fa[u];
        }
        return false;
}

void solve() {
        cin >> n >> m >> a >> b >> c;
        while(m--) {
                int u, v;
                cin >> u >> v;
                g[u].push_back(v);
                g[v].push_back(u);
        }
        tarjan(1);
        dfs(c, -1);
        cout << (check(a) ? "Yes\n" : "No\n");
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```

网络流之最大流

例题

洛谷P3376 【模板】网络最大流——EK算法求最大流

EK 算法一般能处理规模 1000 ~ 10000 的网络。

```cpp
#include <bits/stdc++.h>
#define endl "\n"
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 210, M = 10010;

int n, m, s, t;
int h[N], e[M], ne[M], c[M], idx;
int pre[N];
LL d[N];

void add(int u, int v, int w) {
        e[idx] = v, ne[idx] = h[u], c[idx] = w, h[u] = idx++;
        e[idx] = u, ne[idx] = h[v], c[idx] = 0, h[v] = idx++;
}

bool bfs() {
        vector<int> vis(n + 1, 0);
        queue<int> q;
        q.push(s);
        d[s] = 1e18, vis[s] = 1;
        while(q.size()) {
                int u = q.front();
                q.pop();

                for(int i = h[u]; ~i; i = ne[i]) {
                        int v = e[i];
                        if(!vis[v] && c[i]) {
                                vis[v] = 1;
                                d[v] = min(d[u], 1ll * c[i]);
                                pre[v] = i;
                                if(v == t) return true;
                                q.push(v);
                        }
                }
        }
        return false;
}
```

```
LL EK() {
        LL res = 0;
        while(bfs()) {
                res += d[t];
                for(int i = t; i != s; i = e[pre[i] ^ 1]) {
                        c[pre[i]] -= d[t];
                        c[pre[i] ^ 1] += d[t];
                }
        }
        return res;
}

void solve() {
        cin >> n >> m >> s >> t;
        fill(h + 1, h + 1 + n, -1);
        while(m--) {
                int u, v, w;
                cin >> u >> v >> w;
                add(u, v, w);
        }
        cout << EK() << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) solve();
        return 0;
}
```

洛谷P3376 【模板】网络最大流——Dinic算法求最大流

$Dinic$ 算法一般能处理规模 10000 ~ 100000 的网络。

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 10010, M = 200010;

int n, m, s, t;
int h[N], e[M], ne[M], idx;
int d[N], cur[N];
LL c[M];

void add(int u, int v, int w) {
        e[idx] = v, ne[idx] = h[u], c[idx] = w, h[u] = idx++;
        e[idx] = u, ne[idx] = h[v], c[idx] = 0, h[v] = idx++;
}

bool bfs() {
        queue<int> q;
        q.push(s);
        fill(d + 1, d + 1 + n, -1);
        d[s] = 0, cur[s] = h[s];
        while(q.size()) {
                int u = q.front();
                q.pop();

                for(int i = h[u]; ~i; i = ne[i]) {
                        int v = e[i];
                        if(d[v] == -1 && c[i]) {
                                d[v] = d[u] + 1;
                                cur[v] = h[v];
                                if(v == t) return true;
                                q.push(v);
                        }
                }
        }
        return false;
}

LL find(int u, LL limit) {
```

```
        if(u == t) return limit;

        LL flow = 0;
        for(int i = cur[u]; ~i && flow < limit; i = ne[i]) {
                int v = e[i];
                cur[u] = i;
                if(d[v] == d[u] + 1 && c[i]) {
                        LL f = find(v, min(c[i], limit - flow));
                        if(!f) d[v] = -1;
                        c[i] -= f, c[i ^ 1] += f, flow += f;
                }
        }
        return flow;
}

LL dinic() {
        LL res = 0, f = 0;
        while(bfs()) {
                while(f = find(s, 1e18)) {
                        res += f;
                }
        }
        return res;
}

void solve() {
        cin >> n >> m >> s >> t;
        fill(h + 1, h + 1 + n, -1);
        while(m--) {
                int u, v, w;
                cin >> u >> v >> w;
                add(u, v, w);
        }
        cout << dinic() << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) solve();
```

```
        return 0;
}
```

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
#define sz(x) ((int)x.size())
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int mod = 998244353;
const int N = 210;

// O(V^2E)
int n, m, s, t;
int dep[N], cur[N];
struct edge {
        int v, w, id;
        edge() {}
        edge(int _v, int _w, int _id) : v(_v), w(_w), id(_id) {}
};
vector<edge> e[N];

bool bfs() {
        fill(dep + 1, dep + 1 + n, -1);
        fill(cur + 1, cur + 1 + n, 0);  // 当前弧优化
        queue<int> q;
        q.push(s);
        dep[s] = 0;
        while(sz(q)) {
                int u = q.front();
                q.pop();

                for(auto [v, w, i] : e[u]) {
                        if(w > 0 && dep[v] == -1) {
                                dep[v] = dep[u] + 1;
                                q.push(v);
                        }
                }
        }
        return dep[t] != -1;
}

LL dfs(int u, LL limit) {
```

```
        if(u == t) return limit;

        LL flow = 0;
        for(int i = cur[u]; i < sz(e[u]) && flow < limit; i++) {
                cur[u] = i;             // 当前弧优化
                auto &[v, w, id] = e[u][i];
                if(w > 0 && dep[v] == dep[u] + 1) {
                        LL f = dfs(v, min(1ll * w, limit - flow));
                        w -= f;
                        e[v][id].w += f;
                        flow += f;
                }
        }
        return flow;
}

LL dinic() {
        LL res = 0, f = 0;
        while(bfs()) {
                while(f = dfs(s, 1e18)) {
                        res += f;
                }
        }
        return res;
}

void solve() {
        cin >> n >> m >> s >> t;
        while(m--) {
                int u, v, w;
                cin >> u >> v >> w;
                int x = sz(e[u]), y = sz(e[v]);
                e[u].emplace_back(v, w, y);             // u -> v, 反向边的编号为 y
                e[v].emplace_back(u, 0, x);             // v -> u, 反向边的编号为 x
        }
        cout << dinic() << '\n';
}


int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
```

```
        // cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```

2-SAT

模板

P4782 【模板】2-SAT

```
#include <bits/stdc++.h>
#define x first
#define y second
#define sz(x) ((int)x.size())
using namespace std;

typedef long long ll;
typedef pair<int, int> PII;
const int inf = 0x3f3f3f3f;
const ll INF = 1e18;
const int N = 2000010;

int n, m;
int low[N], dfn[N], timestmp;
int scc_cnt, id[N];
bool in_stk[N];
vector<int> e[N], stk;

void tarjan(int u) {
        dfn[u] = low[u] = ++timestmp;
        stk.push_back(u);
        in_stk[u] = true;
        for(int v : e[u]) {
                if(!dfn[v]) {
                        tarjan(v);
                        low[u] = min(low[u], low[v]);
                } else if(in_stk[v]) {
                        low[u] = min(low[u], dfn[v]);
                }
        }
        if(low[u] == dfn[u]) {
                scc_cnt += 1;
                int cur;
                do {
                        cur = stk.back();
                        stk.pop_back();
```

```
                        in_stk[cur] = false;
                        id[cur] = scc_cnt;
                } while(cur != u);
        }
}

/*
        x:  x
        !x: x + n
*/
void solve() {
        cin >> n >> m;
        while(m--) {
                int i, a, j, b;
                cin >> i >> a >> j >> b;
                if(a && b) {    // a v b <-> !a -> b ^ !b -> a
                        e[i + n].push_back(j);
                        e[j + n].push_back(i);
                } else if(a && !b) {    // a v !b <-> !a -> !b ^ b -> a
                        e[i + n].push_back(j + n);
                        e[j].push_back(i);
                } else if(!a && b) {    // !a v b <-> a -> b ^ !b -> !a
                        e[i].push_back(j);
                        e[j + n].push_back(i + n);
                } else {        // !a v !b <-> a -> !b ^ b -> !a
                        e[i].push_back(j + n);
                        e[j].push_back(i + n);
                }
        }
        for(int i = 1; i <= n * 2; i++) {
                if(!dfn[i]) tarjan(i);
        }
        for(int i = 1; i <= n; i++) {
                if(id[i] == id[i + n]) {
                        cout << "IMPOSSIBLE\n";
                        return;
                }
        }

        cout << "POSSIBLE\n";
        // 编号 scc_cnt -> 1 是拓扑序的顺序
        // !x -> x, 如果 x 的拓扑序大于 !x 的拓扑序, x 赋值 1, 否则赋值 0
        for(int i = 1; i <= n; i++) {
```

```
                cout << (id[i] < id[i + n]) << " \n"[i == n];
        }
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
        // cin >> t;
        while(t--) solve();
        return 0;
}
```

dp

数位dp

例题

[ABC317F] Nim

可以把 limit 和 lead 参数放进状态表示里，用空间换时间。

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
typedef tuple<int, int, int> node;
const int INF = 0x3f3f3f3f;
const int mod = 998244353;
const int N = 65, M = 10, K = 2;

LL f[N][M][M][M][K][K][K][K][K][K];

void solve() {
        LL n, a, b, c;
        cin >> n >> a >> b >> c;
        vector<int> v;
        while(n) {
                v.push_back(n % 2);
                n /= 2;
        }
        n = v.size() - 1;
        memset(f, -1, sizeof f);
        auto dfs = [&](auto &self, int pos,
                                int m1, int m2, int m3,         //remain
                                int l1, int l2, int l3,         //limit
                                int z1, int z2, int z3          //lead
                                ) -> LL {
                if(pos < 0) {
                        return !m1 && !m2 && !m3 && !z1 && !z2 && !z3;
                }
                LL &val = f[pos][m1][m2][m3][l1][l2][l3][z1][z2][z3];
                if(~val) return val;
                val = 0;
                vector<int> up = {
                        l1 ? v[pos] : 1,
                        l2 ? v[pos] : 1,
                        l3 ? v[pos] : 1
                };
                for(int i = 0; i <= up[0]; i++) {
                        for(int j = 0; j <= up[1]; j++) {
                                for(int k = 0; k <= up[2]; k++) {
```

```
                                        int cnt = i + j + k;
                                        if(cnt != 0 && cnt != 2) continue;
                                        LL tmp = self(self, pos - 1,
                                                (m1 * 2 + i) % a, (m2 * 2 + j) % b, (m3 * 2 + k
                                                l1 && i == up[0], l2 && j == up[1], l3 && k == u
                                                z1 && !i, z2 && !j, z3 && !k);
                                        val = (val + tmp) % mod;
                                }
                        }
                }
                return val;
        };
        cout << dfs(dfs, n,
                        0, 0, 0,
                        1, 1, 1,
                        1, 1, 1) << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```

树上背包

例题

洛谷P2014 [CTSC1997] 选课

$f_{u,j,k}$ 表示以 u 为根的子树前 j 个儿子体积不超过 k 的最大学分，转移的时候按照分组背包的方式转移即可。

朴素版时间复杂度：$O(nm^2)$

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;

void solve() {
	int n, m;
	cin >> n >> m;
	vector< vector<int> > e(n + 1);
	vector<int> w(n + 1);
	for(int i = 1, p; i <= n; i++) {
		cin >> p >> w[i];
		e[p].push_back(i);
	}
	vector f(n + 1, vector<int>(m + 2, 0));
	function<void(int)> dfs = [&](int u) {
		f[u][1] = w[u];
		for(int v : e[u]) {
			dfs(v);

			for(int j = m + 1; j >= 1; j--) {
				for(int k = 0; k < j; k++) {
					f[u][j] = max(f[u][j], f[u][j - k] + f[v][k]);
				}
			}
		}
	};
	dfs(0);
	cout << f[0][m + 1] << '\n';
}

int main() {
	ios::sync_with_stdio(false);
	cin.tie(0), cout.tie(0);
	int t = 1;
//	cin >> t;
	while(t--) {
		solve();
	}
```

```
        return 0;
}
```

优化版时间复杂度：$O(n^2)$

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;

void solve() {
	int n, m;
	cin >> n >> m;
	vector< vector<int> > e(n + 1);
	vector<int> w(n + 1);
	for(int i = 1, p; i <= n; i++) {
		cin >> p >> w[i];
		e[p].push_back(i);
	}
	vector f(n + 1, vector<int>(m + 2, 0));
	vector<int> sz(n + 1, 0);
	function<void(int)> dfs = [&](int u) {
		f[u][1] = w[u];
		sz[u] = 1;
		for(int v : e[u]) {
			dfs(v);

			/*
				1. j - k >= sz[u]
				2. j = min(sz[u] + sz[v], m + 1)
				3. k <= sz[v]
			*/
			for(int j = min(m + 1, sz[u] + sz[v]); j >= 1; j--) {
				for(int k = max(0, j - sz[u]); k <= sz[v] && k < j; k++) {
					f[u][j] = max(f[u][j], f[u][j - k] + f[v][k]);
				}
			}
			sz[u] += sz[v];
		}
	};
	dfs(0);
	cout << f[0][m + 1] << '\n';
}
```

```
int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```

SOS(sum over subset)dp

例题

CF449D. Jzzhu and Numbers

f_i 表示序列 a 中 i 的二进制的超集个数，那么 2^{f_i-1} 就表示序列 a 中与和为 i 的超集的集合个数（减去空集）。

再进行容斥，恰好为 0 的集合个数 = 至少有 0 个 1 的集合个数 - 至少有 1 个 1 的集合个数 + 至少有两个 1 的集合个数 - . . .

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = INT_MAX;
const int mod = 1e9 + 7;
const int N = 1 << 20;

LL qmi(LL a, LL b, LL mod) {
        LL res = 1;
        while(b) {
                if(b & 1) res = res * a % mod;
                b >>= 1;
                a = a * a % mod;
        }
        return res % mod;
}

void solve() {
        int n;
        cin >> n;
        vector<int> a(n);
        for(int &x : a) {
                cin >> x;
        }
        vector<LL> f(N, 0), g(N, 0);
        for(int i = 0; i < N; i++) {
                g[i] = __builtin_popcount(i);
        }
        for(int i = 0; i < n; i++) {
                f[a[i]]++;
        }
        for(int j = 0; j < 20; j++) {
                for(int i = 0; i < N; i++) {
                        if(!(i & (1 << j))) {
                                (f[i] += f[i ^ (1 << j)]) %= mod;
                        }
                }
        }
        LL ans = 0;
```

```
        for(int i = 0; i < N; i++) {
                f[i] = (qmi(2, f[i], mod) - 1 + mod) % mod;
                LL val = (g[i] & 1) ? -1 : 1;
                (ans += val * f[i] + mod) %= mod;
        }
        cout << ans << '\n';
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) solve();
        return 0;
}
```

数据结构

二维数点

例题

CF1899G. Unusual Entertainment

主席树写法

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 100010;

int idx, n, q;
int root[N];
struct node {
        int l, r, cnt;
}tr[N * 4 + N * 17];

void clear() {
        idx = 0;
        fill(root + 1, root + 1 + n, 0);
}

int insert(int p, int l, int r, int v) {
        int q = ++idx;
        tr[q] = tr[p];
        if(l == r) {
                tr[q].cnt++;
                return q;
        }
        int mid = (l + r) >> 1;
        if(v <= mid) tr[q].l = insert(tr[p].l, l, mid, v);
        else tr[q].r = insert(tr[p].r, mid + 1, r, v);
        tr[q].cnt = tr[tr[q].l].cnt + tr[tr[q].r].cnt;
        return q;
}

int query(int p, int l, int r, int ql, int qr) {
        if(ql <= l && r <= qr) return tr[p].cnt;
        int res = 0, mid = (l + r) >> 1;
        if(ql <= mid) res += query(tr[p].l, l, mid, ql, qr);
        if(qr > mid) res += query(tr[p].r, mid + 1, r, ql, qr);
        return res;
}
```

```
void solve() {
	cin >> n >> q;
	vector< vector<int> > e(n + 1);
	vector<int> p(n + 1);
	for(int i = 1, u, v; i < n; i++) {
		cin >> u >> v;
		e[u].push_back(v);
		e[v].push_back(u);
	}
	for(int i = 1; i <= n; i++) {
		cin >> p[i];
	}
	int timestmp = 0;
	vector<int> dfn(n + 1), out(n + 1);
	function<void(int, int)> dfs = [&](int u, int fa) {
		dfn[u] = ++timestmp;
		for(int v : e[u]) {
			if(v == fa) continue;
			dfs(v, u);
		}
		out[u] = timestmp;
	};
	dfs(1, -1);
	for(int i = 1; i <= n; i++) {
		root[i] = insert(root[i - 1], 1, n, dfn[p[i]]);
	}
	while(q--) {
		int l, r, x;
		cin >> l >> r >> x;
		int cnt = query(root[r], 1, n, dfn[x], out[x]) -
		query(root[l - 1], 1, n, dfn[x], out[x]);
		cout << (cnt > 0 ? "YES\n" : "NO\n");
	}
	clear();
}

int main() {
	ios::sync_with_stdio(false);
	cin.tie(0), cout.tie(0);
	int t = 1;
	cin >> t;
	while(t--) {
		solve();
```

```
        }
        return 0;
}
```

树状数组离线写法

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
typedef tuple<int, int, int, int> node;
const int INF = 0x3f3f3f3f;

struct bitTree {
        int n;
        vector<int> tr;
        bitTree() {}
        bitTree(int sz) {
                tr.resize(sz + 1);
                n = sz;
                fill(tr.begin(), tr.end(), 0);
        }
        int lowbit(int x) {
                return x & -x;
        }
        void add(int x, int v) {
                while(x <= n) {
                        tr[x] += v;
                        x += lowbit(x);
                }
        }
        int sum(int x) {
                int res = 0;
                while(x) {
                        res += tr[x];
                        x -= lowbit(x);
                }
                return res;
        }
};

/*
  l, r, x
  x: [l, r]
  y: [dfn[x], out[x]]
  (l, dfn[x]) (l, out[x])
```

```
  (r, dfn[x]) (r, out[x])
  l - 1, out[x], -1
  r, dfn[x] - 1, -1
  l - 1, dfn[x] - 1, 1
  r, out[x], 1
 */

void solve() {
        int n, q;
        cin >> n >> q;
        vector< vector<int> > e(n + 1);
        vector<int> p(n + 1);
        for(int i = 1, u, v; i < n; i++) {
                cin >> u >> v;
                e[u].push_back(v);
                e[v].push_back(u);
        }
        for(int i = 1; i <= n; i++) {
                cin >> p[i];
        }
        int timestmp = 0;
        vector<int> dfn(n + 1), out(n + 1);
        function<void(int, int)> dfs = [&](int u, int fa) {
                dfn[u] = ++timestmp;
                for(int v : e[u]) {
                        if(v == fa) continue;
                        dfs(v, u);
                }
                out[u] = timestmp;
        };
        dfs(1, -1);
        vector<node> query;
        for(int i = 0; i < q; i++) {
                int l, r, x;
                cin >> l >> r >> x;
                query.emplace_back(l - 1, out[x], -1, i);
                query.emplace_back(r, dfn[x] - 1, -1, i);
                query.emplace_back(l - 1, dfn[x] - 1, 1, i);
                query.emplace_back(r, out[x], 1, i);
        }
        sort(query.begin(), query.end());
        int idx = 1;
        bitTree tr(n);
```

```
        vector<int> ans(q, 0);
        for(auto [x, y, v, id] : query) {
                while(idx <= x) {
                        tr.add(dfn[p[idx++]], 1);
                }
                ans[id] += tr.sum(y) * v;
        }
        for(int i = 0; i < q; i++) {
                cout << (ans[i] > 0 ? "YES\n" : "NO\n");
        }
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
        cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```

字符串

字符串哈希

例题

ABC331F - Palindrome Query——线段树+字符串哈希

单哈希的冲突概率大约为 $\frac{|S|}{P}$，其中 $|S|$ 为字符串长度，P 为模数，冲突概率较大（本题为 $\frac{10^6}{10^9} = 10^{-3}$）。但是使用五个哈希可以使概率变得非常小，为 10^{-15}。

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;

typedef long long LL;
typedef unsigned long long ull;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 1000010;

int n, q;
string s;
int p[] = {
        1000000007,
        1000000009,
        1000000021,
        1000000033,
        1000000087
};
struct T {
        ull h1, h2, pw;
};
struct node {
        T h[5];
        node() {
                for(int i = 0; i < 5; i++) {
                        h[i] = {0, 0, 1};
                }
        }
        node(char c) {
                for(int i = 0; i < 5; i++) {
                        h[i] = {c, c, p[i]};
                }
        }
}tr[N * 4];

void merge(node &root, node &le, node &ri) {
        for(int i = 0; i < 5; i++) {
                root.h[i].h1 = le.h[i].h1 * ri.h[i].pw + ri.h[i].h1;
                root.h[i].h2 = le.h[i].h2 + ri.h[i].h2 * le.h[i].pw;
                root.h[i].pw = le.h[i].pw * ri.h[i].pw;
        }
```

```
}

void pushup(int u) {
        merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
}

void build(int u, int l, int r) {
        tr[u] = node(s[r]);
        if(l == r) return;
        int mid = (l + r) >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
}

void update(int u, int l, int r, int x, char c) {
        if(x == l && x == r) {
                tr[u] = node(c);
                return;
        }
        int mid = (l + r) >> 1;
        if(x <= mid) update(u << 1, l, mid, x, c);
        else update(u << 1 | 1, mid + 1, r, x, c);
        pushup(u);
}

node query(int u, int l, int r, int L, int R) {
        if(L <= l && r <= R) return tr[u];
        int mid = (l + r) >> 1;
        if(R <= mid) return query(u << 1, l, mid, L, R);
        if(L > mid) return query(u << 1 | 1, mid + 1, r, L, R);
        node res;
        node le = query(u << 1, l, mid, L, R);
        node ri = query(u << 1 | 1, mid + 1, r, L, R);
        merge(res, le, ri);
        return res;
}

void solve() {
        cin >> n >> q >> s;
        s = " " + s;
        build(1, 1, n);
        while(q--) {
```

```
                int op;
                cin >> op;
                if(op == 1) {
                        int x;
                        char c[2];
                        cin >> x >> c;
                        update(1, 1, n, x, *c);
                } else {
                        int l, r;
                        cin >> l >> r;
                        node res = query(1, 1, n, l, r);
                        int flag = 1;
                        for(int i = 0; i < 5; i++) flag &= res.h[i].h1 == res.h[i].h2;
                        cout << (flag ? "Yes\n" : "No\n");
                }
        }
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) solve();
        return 0;
}
```

后缀数组(SA)

```
#include <bits/stdc++.h>
#define x first
#define y second
#define sz(x) ((int)x.size())
using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int mod = 998244353;
const int N = 1000010;

/*
  s[i]: 原串
  x[i]: 离散化后的值
  y[i]: 第二关键字排序数组
  c[i]: 计数数组
  rk[i]: [i, n] 这个后缀的排名
  sa[i]: 排名第 i 的后缀起始坐标
  rk[sa[i]] = i
  height[i]: 排名 i 和 排名 i - 1 的最长公共前缀(lcp)
  h[i] = height[rk[i]], 则 h[i] >= h[i - 1] - 1
 */
int n, m;
char s[N];
int x[N], y[N], rk[N], c[N];
int sa[N], height[N];

void get_sa() {
        m = int('z');
        // 按照第一关键字排序
        for(int i = 1; i <= m; i++) c[i] = 0;
        for(int i = 1; i <= n; i++)     c[x[i] = s[i]]++;
        for(int i = 2; i <= m; i++) c[i] += c[i - 1];
        for(int i = n; i >= 1; i--) sa[c[x[i]]--] = i;

        for(int k = 1; k <= n; k *= 2) {
                // 1. 先按第二关键字排序
                // [n - k + 1, n] 这些后缀没有第二关键字, 肯定排在前面
                int cnt = 0;
```

```
                for(int i = n; i > n - k; i--) y[++cnt] = i;
                for(int i = 1; i <= n; i++) {   // 排名
                        if(sa[i] > k) {         // 第 i 个后缀的第二关键字为第 i + k 个后缀的第一关
                                y[++cnt] = sa[i] - k;
                        }
                }

                // 2. 再按第一关键字排序
                for(int i = 1; i <= m; i++) c[i] = 0;
                for(int i = 1; i <= n; i++) c[x[i]]++;
                for(int i = 2; i <= m; i++) c[i] += c[i - 1];
                for(int i = n; i >= 1; i--) sa[c[x[y[i]]]--] = y[i], y[i] = 0;

                // 3. 离散化 [i, i + 2k]
                // swap(x, y);
                y[sa[1]] = 1, cnt = 1;
                for(int i = 2; i <= n; i++) {
                        y[sa[i]] = (x[sa[i]] == x[sa[i - 1]] &&
                                x[sa[i] + k] == x[sa[i - 1] + k]) ? cnt : ++cnt;
                }
                for(int i = 1; i <= n; i++) x[i] = y[i];
                if(cnt == n) break;
                m = cnt;
        }
}

void get_height() {
        for(int i = 1; i <= n; i++) rk[sa[i]] = i;
        for(int i = 1, k = 0; i <= n; i++) {
                if(rk[i] == 1) continue;
                if(k) k--;
                int j = sa[rk[i] - 1];
                while(i + k <= n && j + k <= n && s[i + k] == s[j + k]) k++;
                height[rk[i]] = k;
        }
}

void solve() {
        cin >> (s + 1);
        n = strlen(s + 1);
        get_sa();
        get_height();
        for(int i = 1; i <= n; i++) {
```

```
                cout << sa[i] << " \n"[i == n];
        }
        for(int i = 1; i <= n; i++) {
                cout << height[i] << " \n"[i == n];
        }
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0), cout.tie(0);
        int t = 1;
//      cin >> t;
        while(t--) {
                solve();
        }
        return 0;
}
```